Software Requirements Specification

for Smart Restaurant Management System - MapMeal

Version: 1.0

Status: Approved

Prepared by: Shayan Nemat, 23K-0899

Syed Waleed Hussain, 23K-0885

Sofia Ayaz, 23K-0807

Organization: FAST University Karachi

Date: 12th December 2025

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to document the full scope, functionalities, and constraints of the **"Smart Restaurant Management System - MealMap"**. This document details the requirements for a dual-role platform (User/Owner) that integrates a Knowledge-Based Recommender System (KB RS) to solve the problem of inefficient restaurant discovery and manual table reservations management. It serves as the primary agreement between the stakeholders and the development team regarding the system's capabilities.

1.2 Document Conventions

- **Must/Shall:** Indicates a mandatory requirement.
- **Should:** Indicates a recommended requirement.
- **User:** Refers to the end-client looking for food.
- **Owner:** Refers to the restaurant administrator.
- **System:** Refers to the MapMeal application and backend.
- **KB RS:** Knowledge-Based Recommender System.

1.3 Intended Audience and Reading Suggestions

- **Project Supervisors:** To evaluate if the project meets the academic complexity standards for DBS, SDA, WebProg and RS courses.
- **Backend Developers:** To understand the business logic required for the Recommender System and Database Normalization rules.
- **Frontend Developers:** To design the UI based on the specific input/output requirements described in Section 3 and 4.
- **QA Testers:** To design test cases for critical flows like "Reservation Booking" and "OTP Authentication".

1.4 Product Scope

MealMap is a web/mobile application designed to bridge the gap between customers and restaurant owners.

- **Current Situation:** Users struggle to find restaurants that match specific criteria (e.g., "Desi food, under 1000 PKR, 4-star rating") without browsing multiple lists. Reservations are largely manual (phone calls).
- **Proposed Solution:** MealMap automates discovery using a Constraint-Based Recommender System. It validates constraints (Cuisine, Price, Rating, Location) against the database to return optimal results. It digitizes the reservation process and provides Owners with a management dashboard to control Menu, Promotions, and Table availability in real-time.

1.5 References

- IEEE Std 830-1998: Recommended Practice for Software Requirements Specifications.
- MealMap Database Schema (SQL).
- UML Design Documents (Use Case, Class, Sequence Diagrams).

---

2. Overall Description

2.1 Product Perspective

MapMeal is a self-contained product that replaces manual restaurant directories. It is designed as a client-server architecture:

- **Frontend:** Mobile/Web Interface for user interaction.
- **Backend:** Python-based Controller logic (SDA + RS implementation).
- **Database:** MySQL relational database (DBS implementation).

2.2 Product Functions

The system provides the following high-level functions:

- **Secure Authentication:** Password hashing and OTP verification for all users.
- **Constraint-Based Search:** Filtering restaurants based on strict user constraints.
- **Interactive Reservation:** Booking tables with real-time status tracking (Pending/Confirmed).
- **Content Management:** Owners can Create, Read, Update, and Delete (CRUD) menus and promotions.
- **Feedback Loop:** Users can rate restaurants, which feeds back into the Recommender System logic.

2.3 User Classes and Characteristics

- **Class 1: The Diner (User)**
    - *Characteristics:* General public, varying tech literacy. Wants quick results.
    - *Key Needs:* Intuitive UI, fast search, confirmation of booking.
- **Class 2: The Manager (Owner)**
    - *Characteristics:* Business owner, uses the system daily.
    - *Key Needs:* Reliability, ability to quickly toggle table availability, clear view of pending requests.
- **Class 3: System Administrator**
    - *Characteristics:* High technical skill.
    - *Key Needs:* Database maintenance, user moderation.

2.4 Operating Environment

- **Client Side:** Android 10+ or iOS 14+; Modern Web Browsers (Chrome, Edge).
- **Server Side:** Application Server (Apache Tomcat or similar).
- **Database:** MySQL 8.0 Community Server.
- **Network:** Requires persistent internet connection (4G/5G/Wi-Fi).

2.5 Design and Implementation Constraints

- **DBMS Constraint:** The database schema must be normalized to at least **3rd Normal Form (3NF)**.
- **SDA Constraint:** The system must follow the **Model-View-Controller (MVC)** design pattern.
- **Hardware:** The system must run on standard commodity hardware; no specialized servers required.
- **Language:** The backend logic must be implemented in OOP compliant languages.

2.6 User Documentation

- **Installation Guide:** For deploying the server and database environment.

2.7 Assumptions and Dependencies

- Users will grant Location Permissions to the application.
- Restaurants will update their "Closed" status manually if an emergency occurs.

---

3. External Interface Requirements

3.1 User Interfaces

- **Login/Signup Screen:** Clean layout requesting Email/Phone. Validation error messages appear in red.
- **Search Dashboard:** Contains dropdowns for "Cuisine", a slider for "Price Range", and star icons for "Minimum Rating".
- **Restaurant Details:** Displays a carousel of food images, a scrollable menu list, and a "Book Now" floating action button.
- **Owner Panel:** A tabular view showing "Manage Menu", "Manage Promotions", and "Reservations". Reservations are color-coded (Yellow=Pending, Green=Confirmed, Red=Cancelled).

3.2 Hardware Interfaces

- **GPS Sensor:** The system polls the device GPS to determine `CurrentLocation` for the distance calculation logic.
- **Network Interface:** Uses the device's Wi-Fi/Cellular radio to communicate with the remote database.

3.3 Software Interfaces

- **MySQL Database:** The system interacts via JDBC (Java Database Connectivity) to execute SQL queries.
- **Google Maps API:** Used to render the restaurant location map on the Details screen.

3.4 Communications Interfaces

- **HTTP/HTTPS:** All data exchange occurs over RESTful API calls using JSON format.
- **Email Verification:** For sending One-Time Passwords (OTP).

---

4. System Features

4.1 System Feature 1: User Registration & Authentication

- **4.1.1 Description and Priority:** High Priority. Ensures only verified users can book tables to prevent spam.
- **4.1.2 Stimulus/Response Sequences:**
  - User enters Name, Email, Password -> Clicks "Sign Up".
  - System validates format -> Sends OTP to Email.
  - User enters OTP -> System creates record in `UserAccount` table.
- **4.1.3 Functional Requirements:**
  - **REQ-1.1:** The system shall validate that the email is unique in the database.
  - **REQ-1.2:** The system shall hash passwords using SHA-256 or bcrypt before storage.
  - **REQ-1.3:** The system shall verify the OTP within a 5-minute validity window.

4.2 System Feature 2: Knowledge-Based Restaurant Search

- **4.2.1 Description and Priority:** High Priority. The core SDA logic of the application.
- **4.2.2 Stimulus/Response Sequences:**
  - User inputs constraints: Cuisine="Chinese", Budget="$$", Rating="4.0+".
  - System constructs a dynamic SQL query joining `Restaurant`, `Cuisine`, and `Rating` tables.

    - System returns a list of restaurants matching *all* constraints.
- **4.2.3 Functional Requirements:**
    - **REQ-2.1:** The system shall automatically detect the user's current city/area.
    - **REQ-2.2:** The Recommender System shall filter results where `Restaurant.MinPrice` <= UserBudget.
    - **REQ-2.3:** The system shall sort results by distance (nearest first) or rating (highest first).

4.3 System Feature 3: Reservation Management

- **4.3.1 Description and Priority:** High Priority. Facilitates the business transaction.
- **4.3.2 Stimulus/Response Sequences:**
    - User selects Date, Time, Guests -> Clicks "Reserve".
    - System saves record with `Status = 'Pending'`.
    - Owner sees notification -> Clicks "Confirm".
    - System updates `Status = 'Confirmed'` -> Notifies User.
- **4.3.3 Functional Requirements:**
    - **REQ-3.1:** The system shall prevent booking for past dates or times.
    - **REQ-3.2:** The system shall require a mandatory "Number of Guests" input (Min: 1, Max: 20).
    - **REQ-3.3:** The Owner shall have the authority to change status to 'Confirmed' or 'Cancelled'.

4.4 System Feature 4: Owner Dashboard (CRUD)

- **4.4.1 Description and Priority:** Medium Priority. Allows owners to keep data fresh.
- **4.4.2 Stimulus/Response Sequences:**
    - Owner selects "Add Menu Item" -> Uploads photo, enters Name/Price.
    - System saves to `MenuItem` and `Photo` tables.
- **4.4.3 Functional Requirements:**
    - **REQ-4.1:** The system shall allow Owners to toggle `IsAvailable` status for individual menu items.
    - **REQ-4.2:** The system shall allow Owners to create temporary Promotions with a `ValidTo` date.

---

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- **Latency:** The Recommender System query must return results in under **3 seconds** on a 4G connection.
- **Scalability:** The database design must support at least **10,000** restaurant records without performance degradation.

5.2 Safety Requirements

- **Data Integrity:** The system must use Transaction Management (Commit/Rollback) to ensure that a Reservation is not partially created if the network fails.
- **Backup:** Automated database backups must occur every 24 hours.

5.3 Security Requirements

- **Access Control:** Only users with `Role='Owner'` can access the Management Dashboard. This must be enforced at the API level (Server-side validation), not just the UI.
- **Encryption:** All data in transit must be encrypted via SSL/TLS.

5.4 Software Quality Attributes

- **Usability:** The interface should follow Material Design guidelines to ensure familiarity for Android users.
- **Maintainability:** The code must be modular (Controllers separate from Database Access Objects) to allow easy updates.

5.5 Business Rules

- **BR-1:** A user cannot have more than 2 "Pending" reservations at the same time to prevent spamming owners.
- **BR-2:** Reviews can only be written by users who have a "Confirmed" and completed reservation history with that restaurant (Future Scope/Optional).

---

6. Other Requirements

Database Requirements

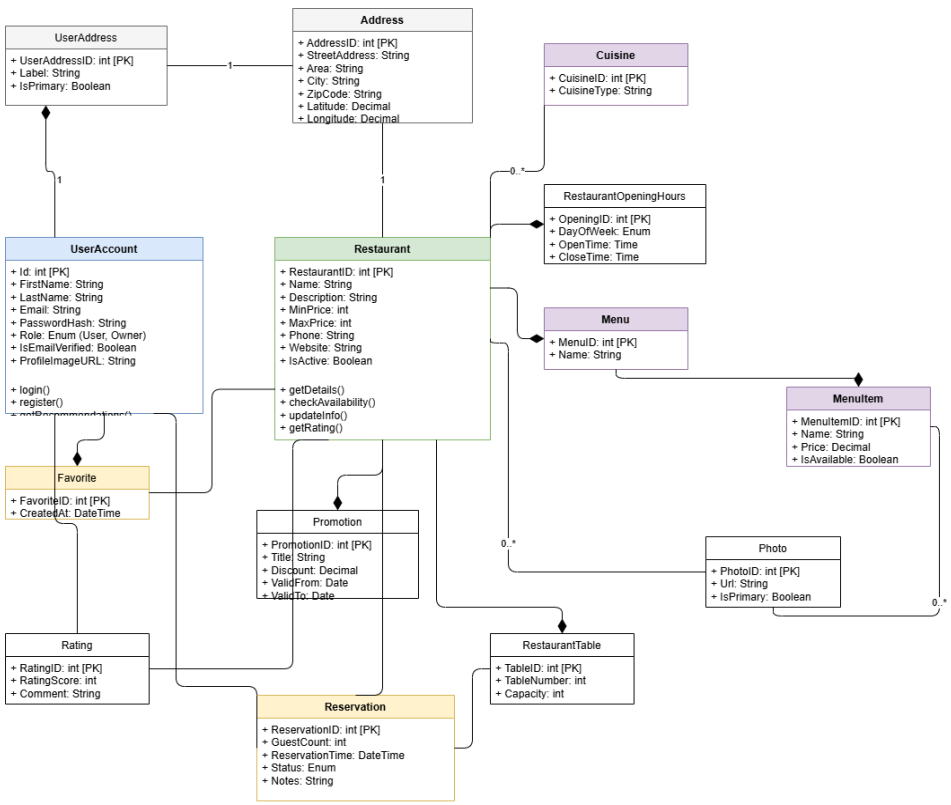
- All tables must utilize proper **Primary Keys** and **Foreign Keys** to maintain referential integrity.
- Indexing must be applied to frequently searched columns (e.g., `CuisineType`, `RestaurantName`, `City`).

Appendix A: Glossary

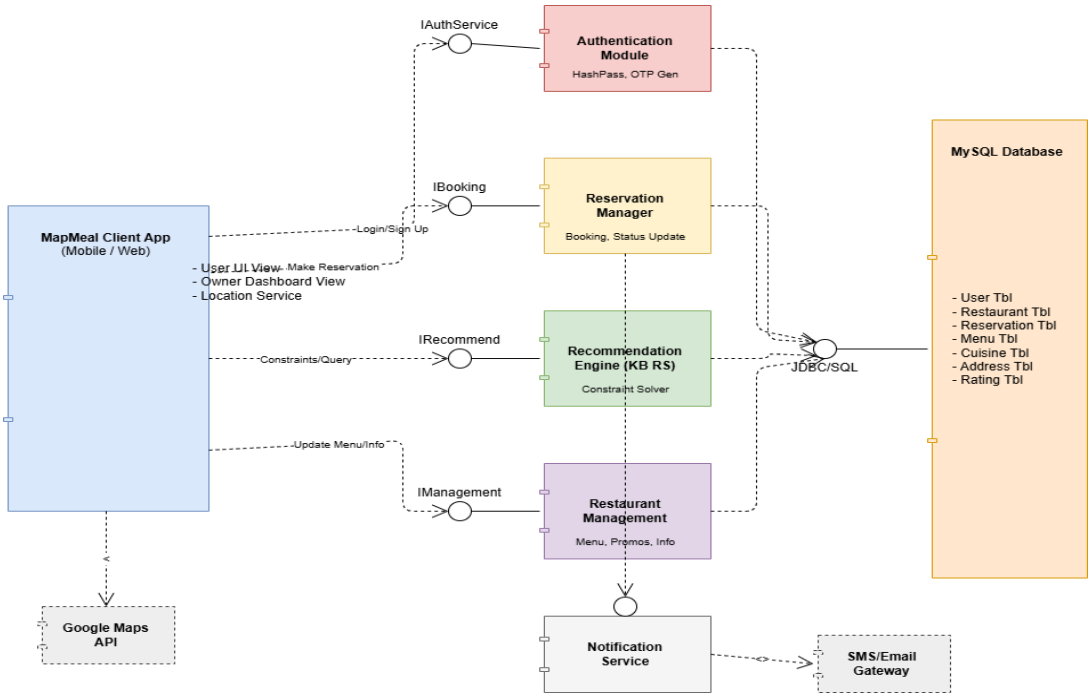
- **SDA:** Software Design and Architecture.
- **KB RS:** Knowledge-Based Recommender System (A logic that recommends items based on specific domain knowledge and user constraints).
- **CRUD:** Create, Read, Update, Delete.
- **OTP:** One-Time Password used for 2-Factor Authentication.

Appendix B: Analysis Models

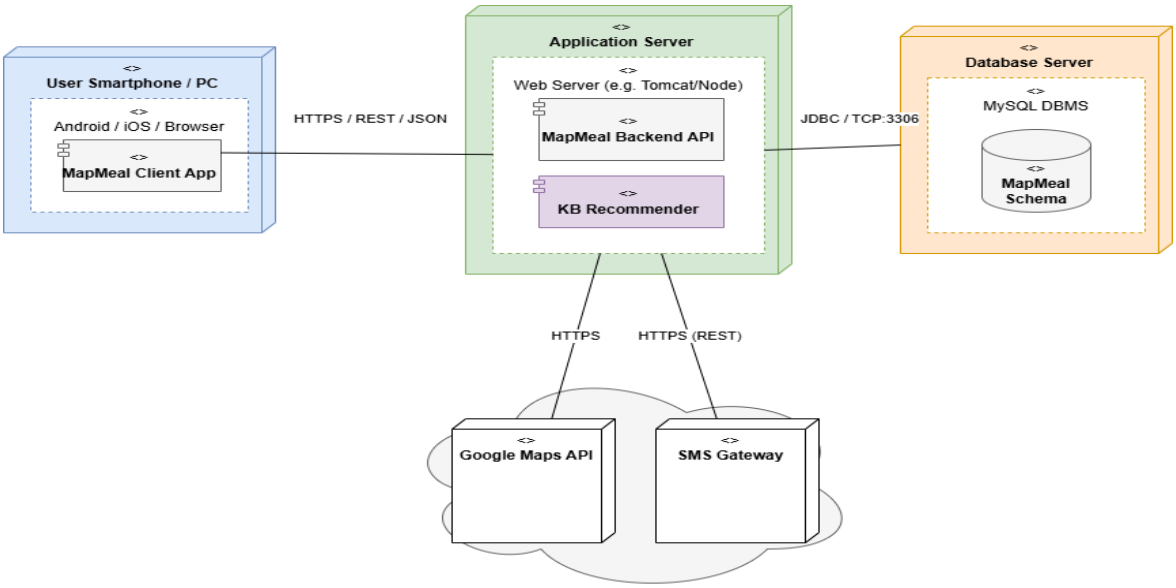
Class Diagram:



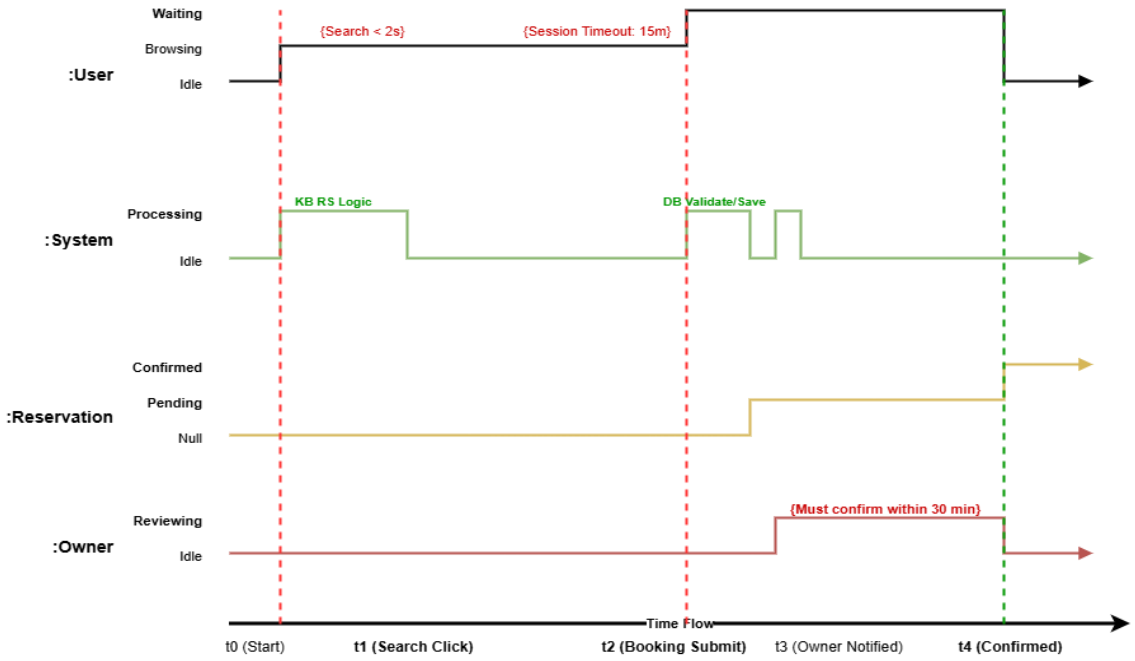
Component diagram:



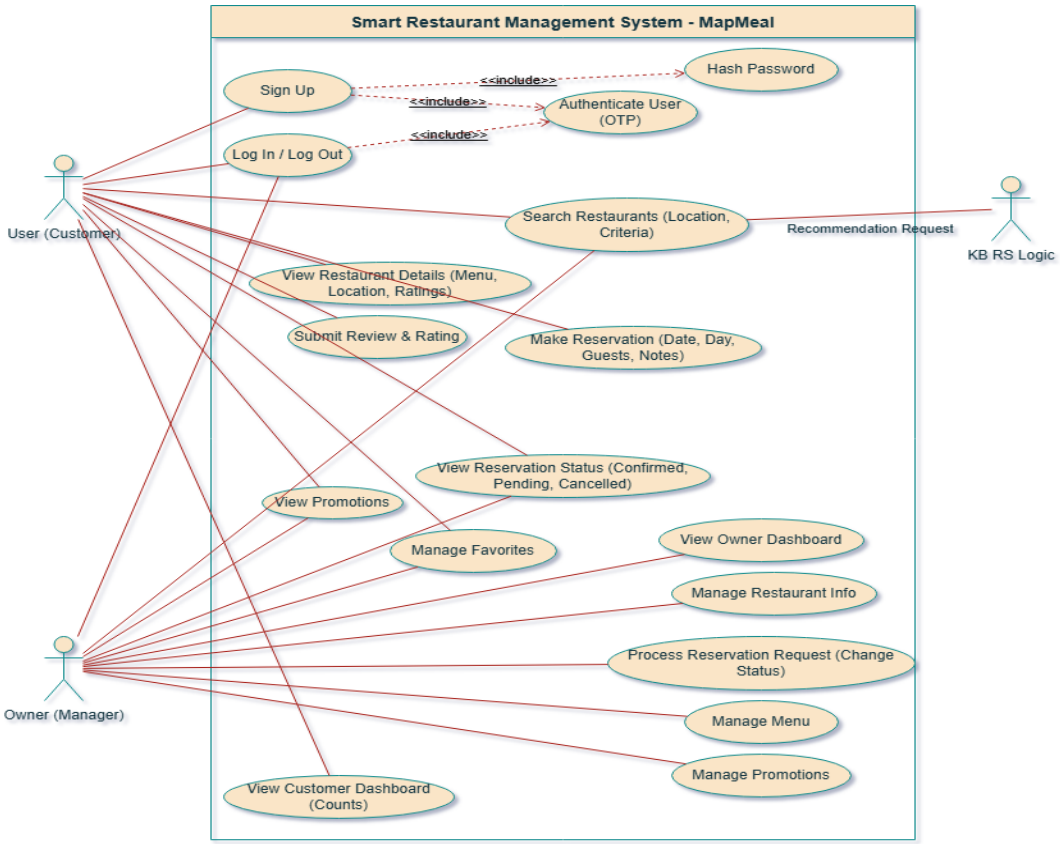
Deployment Diagram:



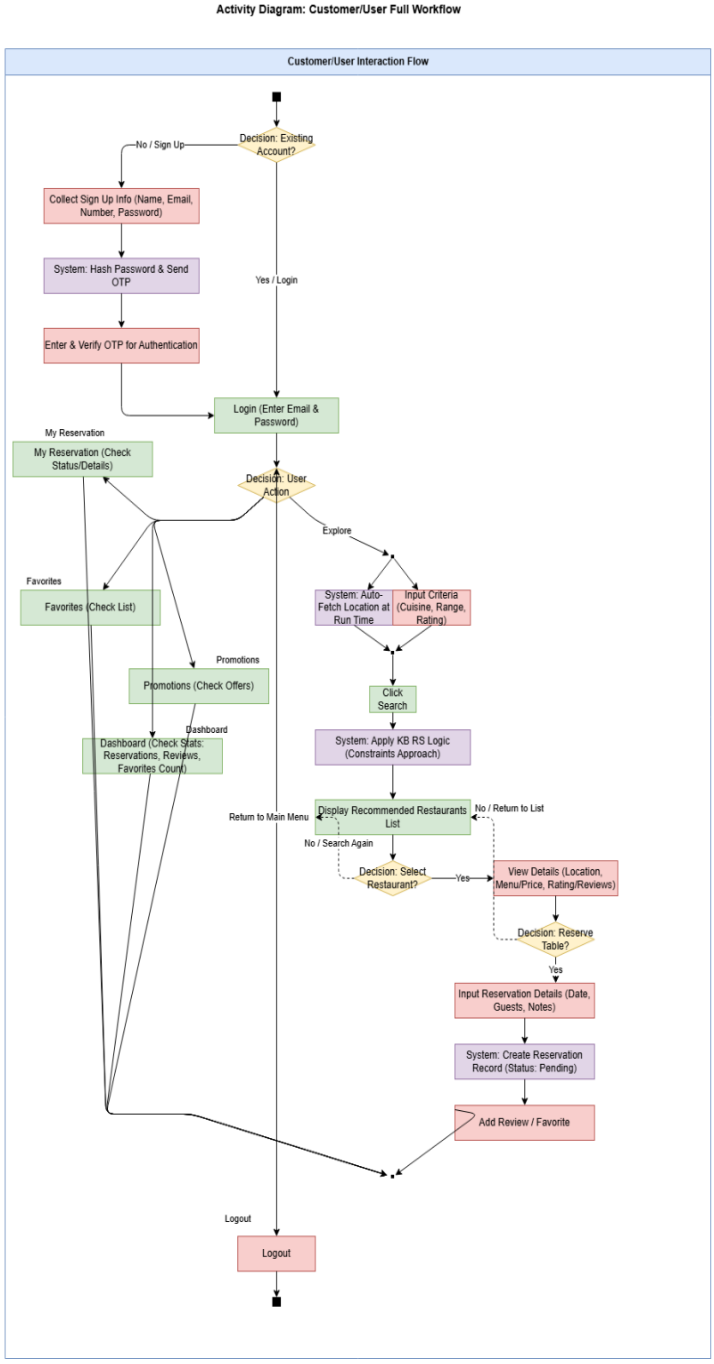
Timing Diagram:



Use-case Diagram:

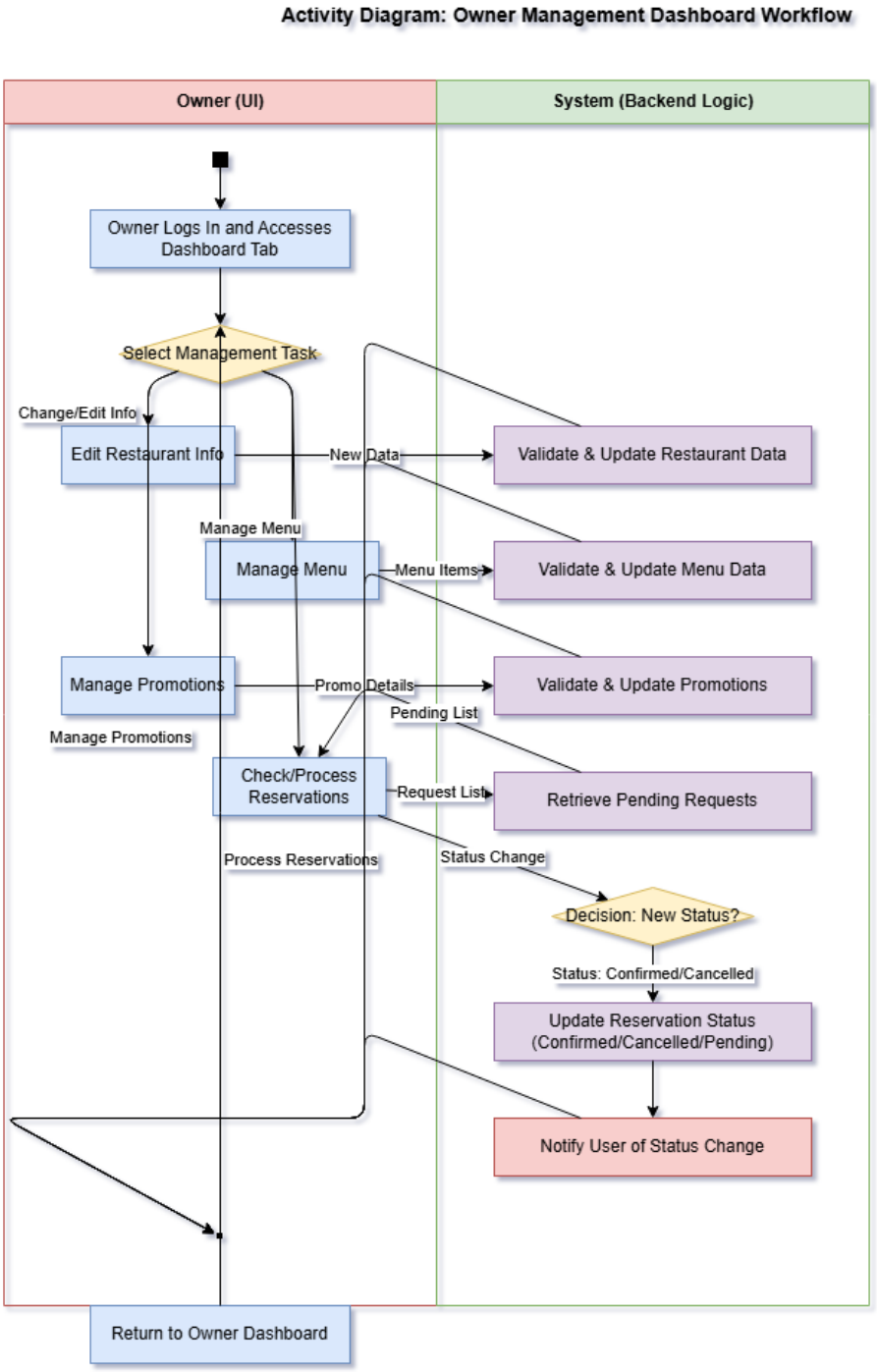


User Activity Diagram:

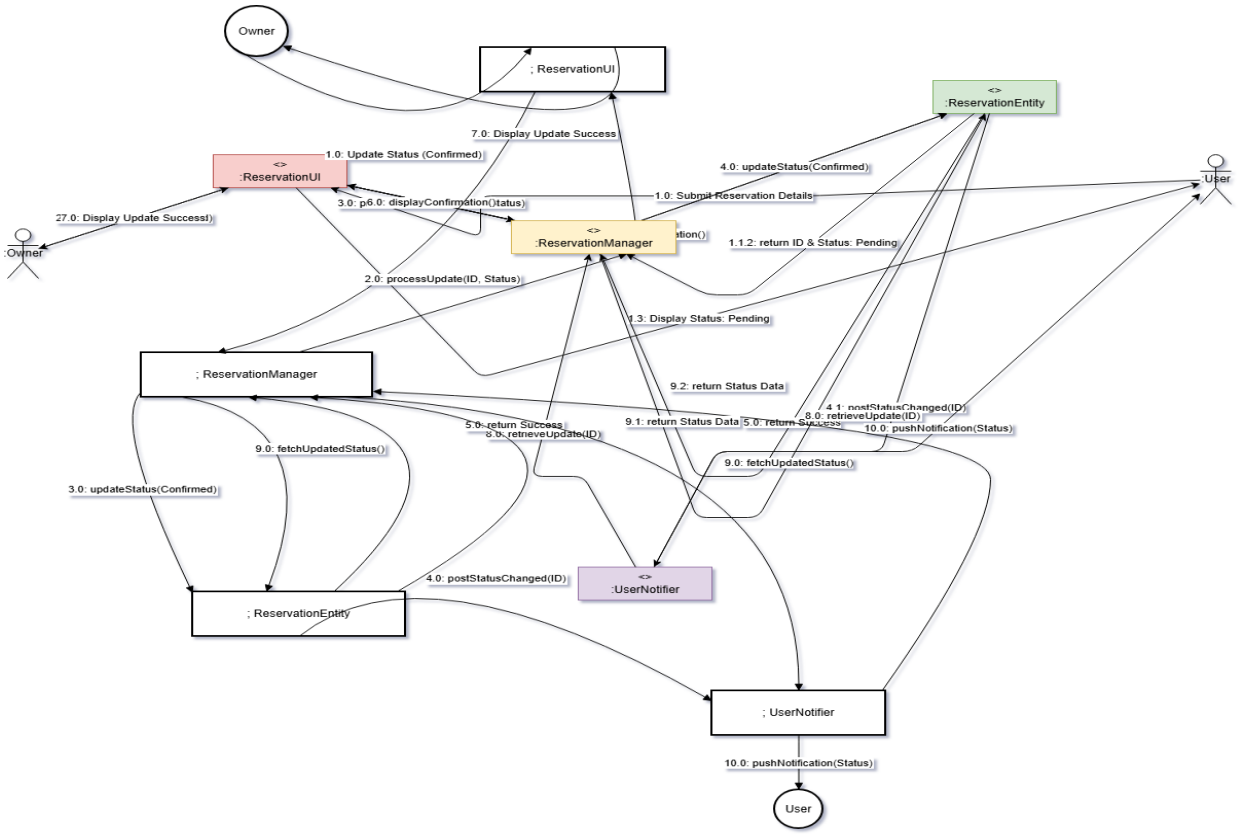


Explore Restaurants

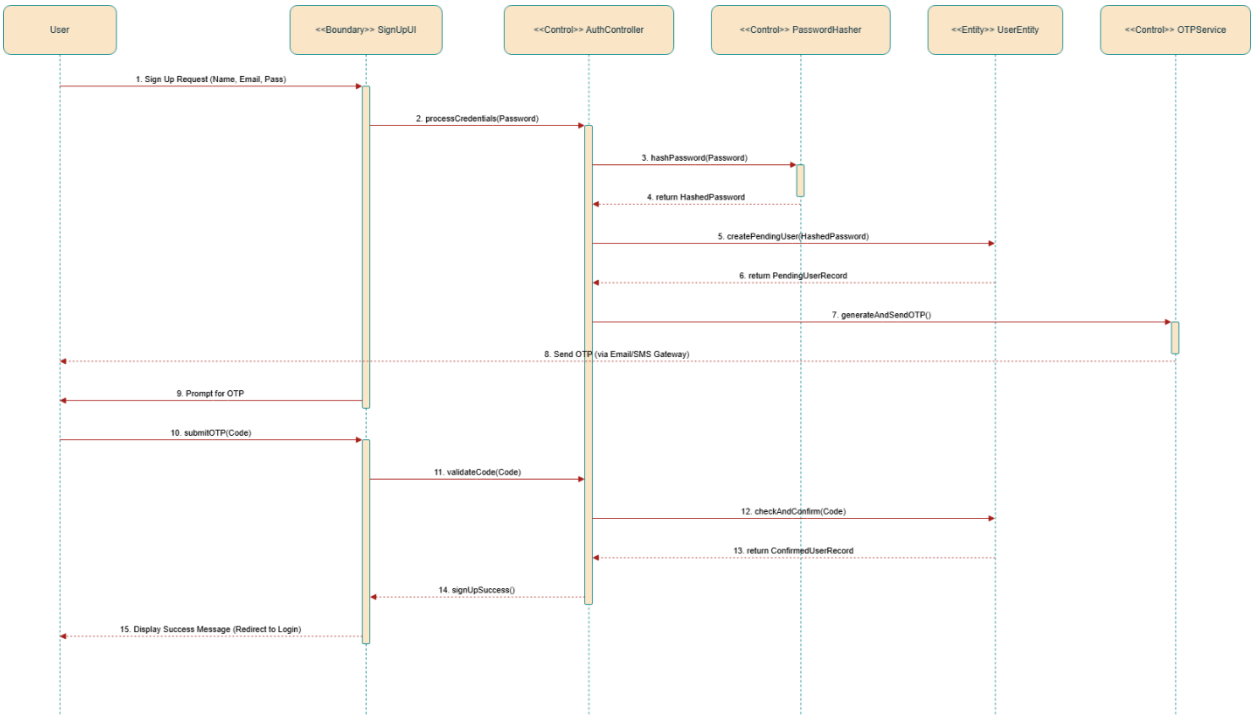
Owner Activity Diagram:



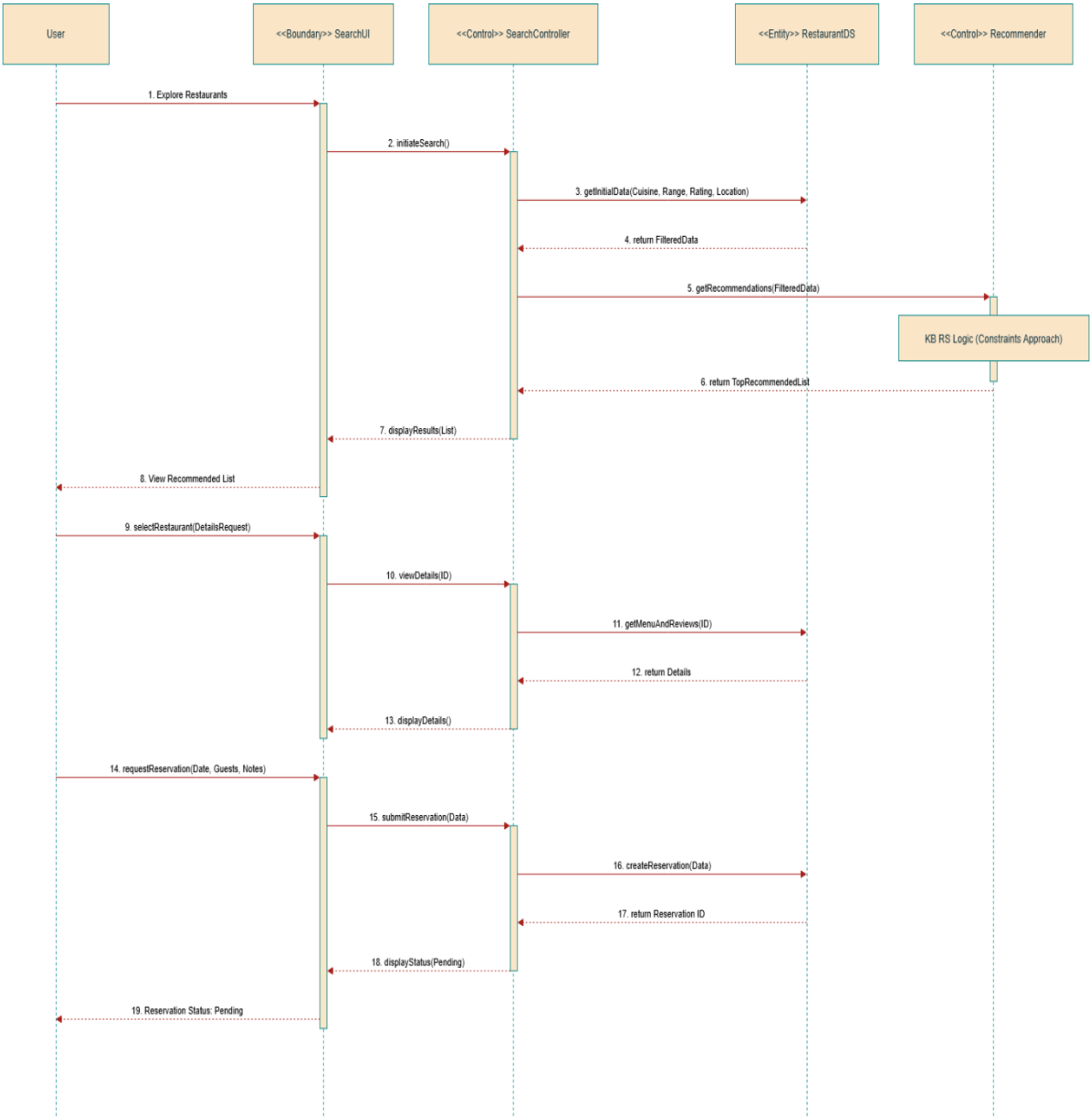
Collaborative Diagram:



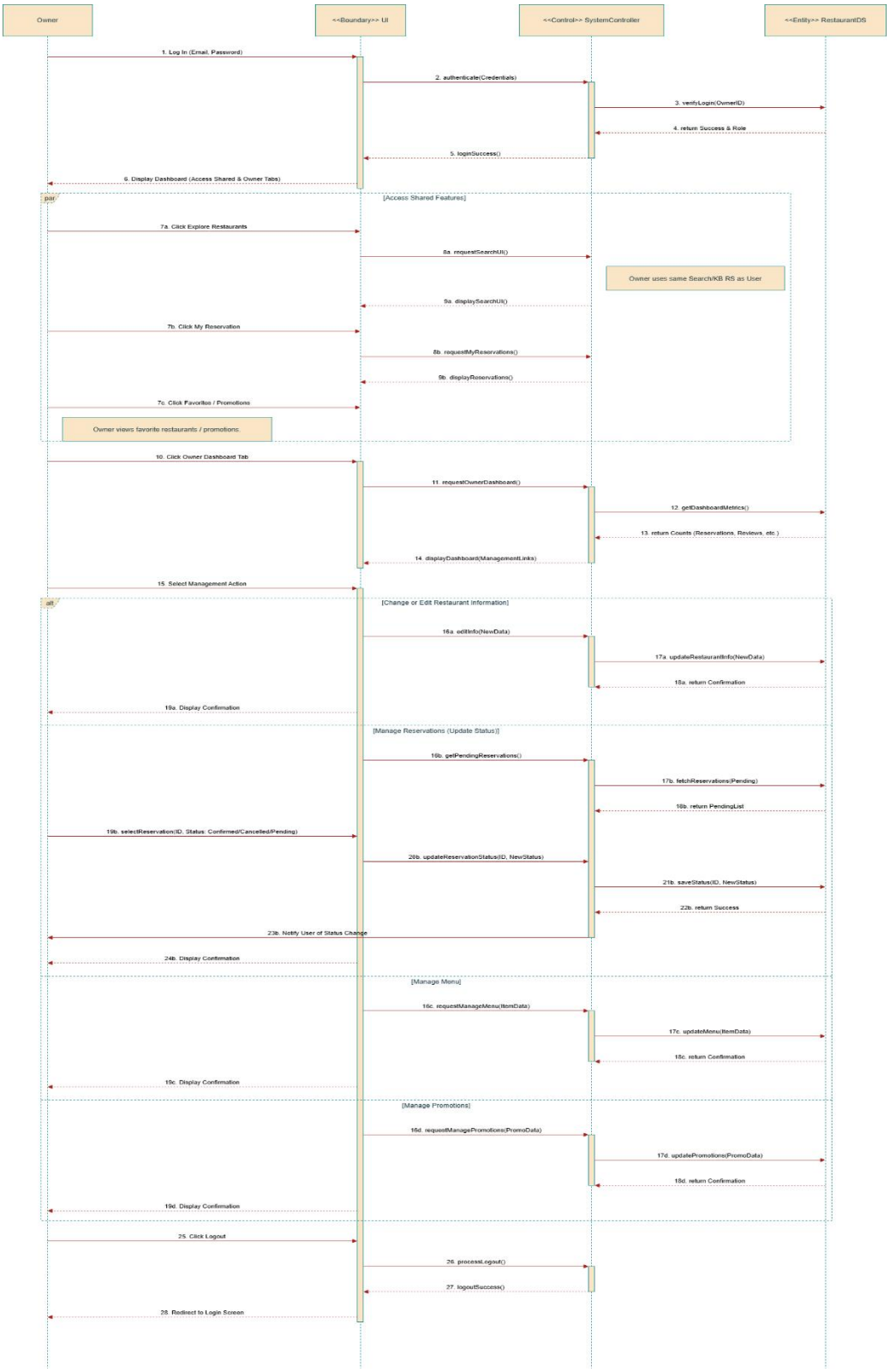
Sign-up Sequence Diagram:



User Sequence Diagram:



Owner Sequence Diagram:



Appendix C: To Be Determined List

- **TBD-1:** Integration with a Payment Gateway (Stripe/JazzCash) for upfront deposits.
- **TBD-2:** "Waitlist" feature for fully booked restaurants.